**SQL 집약과 자르기 – 집합의 세계**

* SQL의 특징적인 사고방식 중에, **레코드** 단위가 아닌 **레코드의 ‘집합’** 단위로 처리를 기술하는 것이 있다.
* 이런 사용방식을 **집합 지향**이라고 부른다.
* **GROUP BY, HAVING** 구와 이것과 함께 사용하는 **SUM 또는 COUNT 등의 집약 함수**의 사용이 대표적인 것들이다.

1. **집약**

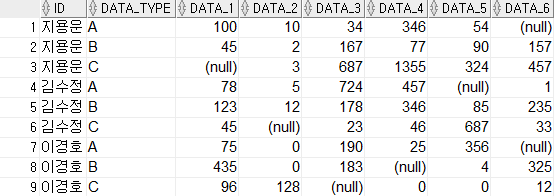
* 집약 함수 : COUNT, SUM, AVG, MAX, MIN
* 이 외에 집약함수들도 있지만, 표준 SQL에 있는 집약함수는 5개뿐 이다.

1. **여러 개의 레코드를 한 개의 레코드로 집약**

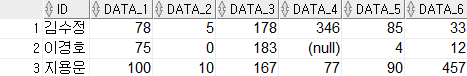
* 집약의 결과를 살펴보기 위해 간단한 예제를 살펴본다.
* data\_1 ~ 6 필드는 사람에 대해서 무언가를 나타내는 정보다.
* data\_type이 A면 data\_1, data\_2 // B면 data\_3, data\_4, data\_5 // C면 data\_6 의 데이터를 사용한다.

-- 비집약 테이블

select \* from NonAggTbl;



--  문제] 위와 같이 한 사람에 대한 필요한 정보가 여러 개의 레코드에 분산되어 있을 때, 다음의 테이블과 같이 사람마다 필요한 정보만 집약한 테이블을 만들어라.



-- 정답

select id,

max(case when data\_type='A' then data\_1 else null end) as data\_1,

max(case when data\_type='A' then data\_2 else null end) as data\_2,

max(case when data\_type='B' then data\_3 else null end) as data\_3,

max(case when data\_type='B' then data\_4 else null end) as data\_4,

max(case when data\_type='B' then data\_5 else null end) as data\_5,

max(case when data\_type='C' then data\_6 else null end) as data\_6

from NonAggTbl

group by id;

* CASE 식과 GROUP BY 식을 응용
* GROUP BY 구로 집약했을 때, SELECT 구에 입력할 수 있는 것은 3가지다.
  + 상수
  + GROUP BY 구에서 사용한 집약 키
  + 집약함수

-- 위에서 사용한 쿼리 (집약 전)

select id,

case when data\_type='A' then data\_1 else null end as data\_1,

case when data\_type='A' then data\_2 else null end as data\_2,

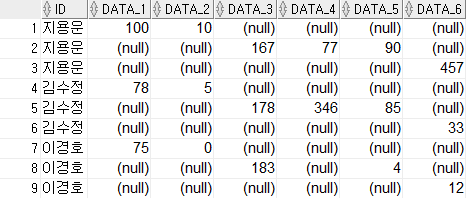
case when data\_type='B' then data\_3 else null end as data\_3,

case when data\_type='B' then data\_4 else null end as data\_4,

case when data\_type='B' then data\_5 else null end as data\_5,

case when data\_type='C' then data\_6 else null end as data\_6

from NonAggTbl;



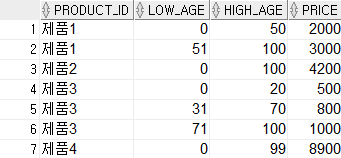
* GROUP BY로 데이터를 자르는 시점에는 각 집합에 3개의 요소가 있다.
* 그런데 여기서 집약함수가 적용되면 NULL을 제외하고 하나의 요소만 있는 집합이 만들어 진다.

1. **합쳐서 하나**

**# 문제 1**

-- 제품의 대상 연령별 가격을 관리하는 테이블.

select \* from PriceByAge;



-- 문제1] 이런 제품 중 0~100세까지 모든 연령이 가지고 놀 수 있는 제품을 구해라.

-- 아래와 같은 테이블이 나와야 한다.

-- 제품3의 경우, 21~30세 연령이 빠지므로 모든 연령에 해당 X.



-- 정답]

select product\_id

from PriceByAge

group by product\_id

having sum(high\_age - low\_age +1 ) = 101;

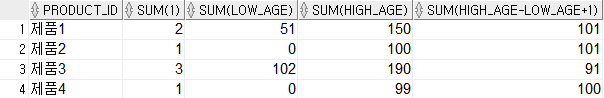
* 일단 집약 단위가 제품이므로 집약 키는 제품 ID로 한다.
* 이어서 각 범위에 있는 상수 개수를 모두 더한 합계가 101인 제품을 선택한다.
* 0~100까지 이므로 사이에 있는 상수의 개수는 101개라는 것에 주의!
* HAVING 구의 ‘high\_age - low\_age +1’로 각 레코드의 연령 범위에 있는 정수 개수를 구한다.

-- 참고, 정답에서 사용한 각 필드

select product\_id, sum(1), sum(low\_age), sum(high\_age), sum(high\_age - low\_age + 1)

from PriceByAge

group by product\_id;

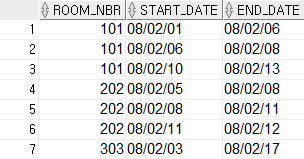


**# 문제 2**

-- 문제2] 아래 테이블에서 사람들이 숙박한 날이 10일 이상인 방을 선택한다.

-- 숙박한 날의 수는 도착일이 2월 1일, 출발일이 2월 6일이라면 5박이므로 5일 이다.

select \* from HotelRooms;



-- 정답

select room\_nbr, sum(end\_date - start\_date) as working\_days

from HotelRooms

group by room\_nbr

having sum(end\_date - start\_date) >=10;



1. **자르기**

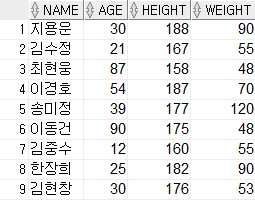
* 지금까지 **GROUP BY** 구에 대해 레코드의 ‘**집약**’이라는 측면을 강조했다.
* '**자르기**’ 라는 중요한 기능이 하나 더 있다.
* “한 개의 구에 두 개의 연산이 들어 있다는 것은 GROUP BY 구에 대한 이해를 막는 원인이 된다.”

1. **자르기와 파티션**

**# 문제 1**

-- 문제1] 이름 첫 글자를 사용해 특정한 글자로 시작하는 이름을 가진 사람이 몇 명인지 집계해라.

select \* from Persons;



-- 정답

select substr(name, 1, 1) as label , count(\*)

from Persons

group by substr(name, 1, 1);



**# 문제 1-2**

--문제 1-2] 나이를 기준으로 어린이(20세미만), 성인(20~69세), 노인(70세 이상)으로 나눠라

-- 출력: 나이구분, 명수



-- 정답]

select case when age<20 then '어린이'

when age between 20 and 69 then '성인'

when age>=70 then '노인'

else null end as age\_class,

count(\*)

from persons

group by case when age<20 then '어린이'

when age between 20 and 69 then '성인'

when age>=70 then '노인'

else null end;

* 자르기의 기준이 되는 키를 GROUP BY 구와 SELECT 구 모두에 입력하는 것이 포인트.
* PostgreSQL, MySQL 에서는 SELECT 구에 붙인 ‘age\_class’라는 별칭을 사용해 ‘GROUP BY age\_class’처럼 단순하게 작성할 수도 있다.

**# 문제 1-3**

-- 문제1-3] BMI 수치를 바탕으로 저체중(18.5 미만), 정상(18.5 이상 25 미만), 과체중(25 이상)으로 분류해라.

-- BMI = 체중 / { (키/100)^2 }

-- 출력: 이름, BMI, 분류

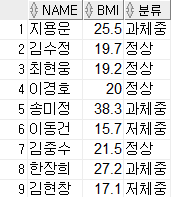
select name, round(weight/power(height/100, 2), 1) as BMI,

case when weight/power(height/100, 2) <18.5 then '저체중'

when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25 then '정상'

when weight/power(height/100, 2) >=25 then '과체중' else null end as 분류

from persons;



**# 문제 1-4**

-- 문제 1-4] 저체중, 정상, 과체중으로 분류하고 명수를 구하라

-- 출력: 체중분류, 명수

select

case when weight/power(height/100, 2) <18.5 then '저체중'

when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25 then '정상'

when weight/power(height/100, 2) >=25 then '과체중' else null end as 분류,

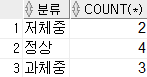
count(\*)

from persons

group by case when weight/power(height/100, 2) <18.5 then '저체중'

when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25 then '정상'

when weight/power(height/100, 2) >=25 then '과체중' else null end;

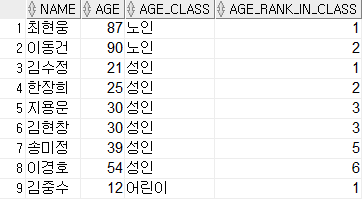


1. **PARTITION BY 구를 사용한 자르기**

* ‘GROUP BY’ 구에서 **집약 기능을 제외하고** **자르는 기능만 남긴 것이** 윈도우 함수의 ‘**PARTITION BY**’ 구.
* 실제로 집약이라는 기능을 제외하면 두 구문은 실질적인 기능에 차이가 없다.
* 한마디로 PARTITION BY 구를 사용해도 단순히 필드 이름뿐만 아니라 CASE식, 계산식을 사용한 **복잡한 기준을 사용할 수 있다는 말** 이다.

-- 문제] 이전에 살펴 봤던 연령 범위 테이블(Persons)에 파티션 자르기를 사용해보자.

-- 출력: 이름, 나이, age\_class, age\_class별 나이 등수



-- 정답]

select name,

age,

case when age<20 then '어린이'

when age between 20 and 69 then '성인'

when age>=70 then '노인'

else null end as age\_class,

rank() over(partition by

case when age<20 then '어린이'

when age between 20 and 69 then '성인'

when age>=70 then '노인'

else null end

order by age) as age\_rank\_in\_class

from persons

order by age\_class, age\_rank\_in\_class;

* 마지막에 있는 age\_rank\_in\_class가 각 파티션 내부에서의 나이 순위를 나타내는 필드.
* PARTITION BY 구는 GROUP BY 구와 달리 집약 기능이 없으므로, 원래 Persons 테이블의 레코드가 모두 원래 형태로 나오는 것을 주목.
* PARTITION BY 구는 입력에 정보를 추가할 뿐이므로 원본 테이블에 정보를 완전히 유지한다.